Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мережних технологій факультету інформатики

Розробка чат-бота для Факультету Інформатики з можливостями NLP
Текстова частина до курсової роботи

Галузь знань: 12 «Інформаційні технології»
Спеціальність: 122 «Комп'ютерні науки та інформаційні технології »

Керівник курсової роботи

доктор техн. наук, доц. Глибовець А.М
(підпис)

"____" _____ 2020 р.

Виконав студент КНІТ-4

Мороз А.В

"____" _____ 2020 р.

Київ 2020

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ

Зав. кафедри мережних технологій,

проф., д.ф.-м.н.
_____Г. І. Малашонок

(підпис)

„____”_____2020 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студенту Морозу Андрію Вікторовичу факультету інформатики 4 курсу

ТЕМА Розробка чат-боту для Факультету Інформатики

Вихідні дані:

Зміст ТЧ до курсової роботи:

Індивідуальне завдання

Календарний план

Анотація

Вступ

РОЗДІЛ 1: Вступ до NLP

РОЗДІЛ 2: Чат-боти

РОЗДІЛ 3: Вибір інструментів розробки

РОЗДІЛ 4: Практична частина

Висновки

Список використаної літератури

Тема: Проектування та розробка високонавантажених застосувань

Календарний план виконання роботи:

№ п/п	Назва етапу дипломного проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання теми курсової роботи.	16.10.2019	
2.	Пошук тематичної літератури	29.10.2019	
3.	Ознайомлення з тематичними матеріалами та побудова структури практичної та теоретичної частин курсової роботи	15.11.2019	
4.	Написання вступу та змісту роботи	11.12.2019	
5.	Ознайомлення з обробкою природних мов та написання першої частини	22.01.2020	
6.	Ознайомлення з чат-ботами та написання другого розділу	10.02.2020	
7.	Вибір інструментів розробки та написання третього розділу	26.02.2020	
7.	Розробка чат-боту	02.03.2020	
8.	Написання четвертого розділу	15.03.2020	
9.	Оформлення роботи відповідно до вимог написання курсової роботи.	27.03.2020	
10.	Створення презентації та написання доповіді для захисту курсової роботи.	28.03.2020	
11.	Узгодження попередньої версії роботи з керівником	02.04.2020	
12.	Внесення змін до курсової роботи відповідно до зауважень наукового керівника	05.04.2020	
13.	Захист курсової роботи	Квітень 2020	

Студент Мороз А.В
Керівник Глибовець А.М.
“_____”_____

Анотація

У даній роботі розбираються основні поняття NLP(natural language processing). Сфери використання NLP та його сучасний розвиток. Розбираються когнітивні сервіси які дозволяють використовувати NLP швидко та зручно, розглядаються переваги так недоліки сервісів. Розбираються відомі інструменти розробки чат-ботів для Telegram, а також описується процес побудови чат-бота з можливостями NLP

Зміст

Вступ

З моменту створення інтернету людство генерує неймовірні обсяги інформації. Кожен рік обсяги інформації збільшуються й на даний момент оцінується що людство в день генерує більше 2,5 квінтильйони байт даних.

Велика частина інформації що генерується є дані створенні під час спілкування людини. Це можуть бути коментарі написані у соціальній мережі, електронні листи, відео трансляції, голосові повідомлення, відгуки клієнтів або запити щодо підтримки клієнтів тощо. Дані згенеровані природною мовою людини можуть відрізнятися не тільки за типом, проте й за мовою написання. Оскільки, що ми виражаємо (усно чи письмово), містить величезну кількість інформації. Тема, яку ми обираємо, наш тон, наш підбір слів, все додає певного типу інформації, яку можна інтерпретувати і з неї можна витягти основну ідею. Теоретично ми можемо зрозуміти і навіть передбачити поведінку людини, використовуючи цю інформацію. Але є проблема: одна людина може генерувати сотні чи тисячі слів, а кожне речення з може бути з відповідною складністю та структурою. Якщо ви хочете масштабно проаналізувати кілька сотень, тисяч чи мільйонів людей та дізнатися основною інформацію з їх складених речень у певному регіон, то ситуація не може бути неймовірно важкою для виконання.

Саме для таких задач й ідеально підходить natural language processing. NLP являє собою автоматичне обробку природньої людської мови, усної чи текстової. Кожен день ми зустрічаєм реальну цінність використання цієї технології, хоча можливо й не помічаємо її прямого впливу на нас.

1. Вступ до NLP

1.1 Що таке NLP

Natural language processing (NLP) - це область інформатики та штучного інтелекту, що стосується взаємодії між комп'ютерами та природними мовами, зокрема, як програмувати комп'ютери, щоб він якісно міг обробити велику кількість природних мовних даних.

Простіше кажучи, обробка природних мов - це здатність комп'ютерів розуміти людську мову під час її розмови. NLP допомагає якісно та швидко аналізувати, розуміти та виводити значення з людської мови.

1.2 Проблеми даних за якими стикається обробка природних мов

Дані, отримані в результаті розмов, повідомлень або новин, є прикладами неструктурованих даних. Неструктуровані дані не вписуються в традиційну структуру рядків і стовпців реляційних баз даних, проте представляють переважну більшість даних, наявних у реальному світі. Вони не структуровані та такими даними важко маніпулювати. Тим не менше, завдяки прогресу в таких дисциплінах, як машинне навчання, відбувається велика революція. Сьогодні мова йде не про спробу інтерпретації тексту чи мовлення на основі його ключових слів, а про розуміння значення цих слів. Таким чином, можна виявити прийоми мови, такі як іронія, або навіть дізнатися настрій людини під час написання тексту.

1.3 Що включає в себе NLP

Обробка природних мов передбачає два основних напрямки роботи: Розуміння природної мови(NLU) та генерація природних мов (NLG). Якщо ваша задача є те щоб комп'ютер міг зрозуміти значення або прихований сенс будь-яких людських мови то саме розуміння природної мови (NLU) намагається зрозуміти зміст написаного тексту. Він намагається виділити основну структуру яка була закладена у всьому реченні так кожному слові. Наприклад когнітивний сервіс LUIS від Microsoft Azure чудово показує, як діє розуміння природних мов. Якщо ж ваша задача складається з того щоб комп'ютер міг граматично правильно та зрозуміло відповісти на дані які йому надійшли, при тому щоб згенерований текст було складно відрізнити від того що напише людина, тоді вам потрібна генерація природних мов. З нею комп’ютер повинен з оглядом на заданий необроблений текст повинен створити граматично правильний текст і підлаштовується під контекст та певний стиль. Чудовим прикладом NLG є IBM Watson NLG.

NLU та NLG залучають різні види досліджень, тому деякі компанії більша кількість компанії спеціалізується тільки на одній с з цих сфер Щоб коротко підсумувати NLU "читає", а NLG "пише".

1.4 Що потрібно для NLG

Основна вимога для генерації природної мови - це володіння або доступ до даних. Для того, щоб будь-яке програмне забезпечення для генерації природних мов створювало подібний до людського текст, то формат змісту повинен бути наданий моделі (за допомогою шаблонів, робочих процесів, заснованих на правилах, та підходів, орієнтованих на наміри), а

потім подавати дані, з яких створюється структуровані дані та зрозумілі для комп’ютера дані.

1.5 Стадії процесів в NLG

У спробах імітувати людське мовлення системи NLG використовували різні методи та хитрощі, щоб адаптувати свій стиль письма, тон та структуру відповідно до аудиторії, контексту та меті розповіді. У 2000 р. Архітектура NLG від Рейтера та Дейла виділила три стадії процесу NLG:

- Планування документів
- Мікропланування
- Реалізація

Планування документів: вирішення того, що слід говорити, та створення абстрактного документа, який окреслює структуру інформації поданої інформації.

Мікропланування: генерація виразів які посилаються на інші вирази, вибір конкретних слів та агрегація для конкретизації специфіки документа з яким працюють.

Реалізація: перетворення специфікацій абстрактних документів у реальний текст, використовуючи основні знання про синтаксис, морфологію тощо.

Ці три стадії демонструють основні етапи генерації природних мов, однак конкретні кроки та підходи, а також моделі, що застосовуються, можуть значно відрізнятись із розвитком технології. Є два основні підходи до генерування мови: використання шаблонів та динамічне створення

документів. Хоча лише остання вважається "справжньою" NLG. Існували довгі й багатоступеневий шляхи від основних простих шаблонів до найсучасніших, і кожен новий підхід розширював функціональність та додав мовних можливостей.

1.6 Типи підходів до NLG

1.6.1 Простий підхід із заповненням пропусків

Один з найстаріших підходів - проста система шаблонів із заповненням пропусків. У текстах, які мають заздалегідь задану структуру та потребують лише невеликого обсягу даних для заповнення, цей підхід повинен автоматично заповнювати такі розриви потрібними даними, отриманими з рядків електронних таблиць, записів таблиці бази даних тощо. Загалом, ви можете змінювати певні аспекти тексту: наприклад, ви можете вирішити, чи потрібно писати числа чи залишити їх, такий підхід є досить обмеженим у використанні та не вважається "справжнім" NLG.

1.6.2 Скрипт або бізнес правила

Основні системи заповнення пропусків були розширені за допомогою загально використовуваних програмних структур та скриптових мов або за допомогою бізнес-правил. Скриптовий підхід, такий як використання веб-шаблонізаторів, вбудовував шаблон у різні мови програмування загального призначення, що й дозволяло створювати складні умови, цикли, отримувати доступи до бібліотек та інші переваги мов загального призначення. Бізнес правила, які й використовуються більшістю інструментів для складання документів , працюють аналогічно, але зосереджуються на написанні бізнес правил, а не на скриптах. Хоча цей

метод є більш потужними, ніж просто заповнення пропусків, такі системи все ще не мають повноцінних лінгвістичних можливостей і не можуть надійно створити складні високоякісні тексти.

1.6.3 Граматичні функції на словесному рівні

Розробка систем на основі шаблонів - додавання граматичних функцій на рівні слів для роботи з морфологією та орфографією, а також для обробки можливих винятків. Ці функції полегшували генерацію граматично правильних текстів та написання складних систем шаблонів.

1.6.4 Динамічна генерація речень

Нарешті, зробивши кроки в розробці від шаблонного підходу вже до динамічного NLG. Цей підхід динамічно створює речення з уявлень про значення, яке передається реченням та / або його бажаною мовною структурою. Динамічне створення означає, що система може генерувати зрозумілі відповіді в нетипових випадках, не вимагаючи від розробника явно писати код для кожного можливого випадку. Це також дозволяє системі лінгвістично "оптимізувати" речення різними способами, включаючи посилання, агрегування, упорядкування та сполучення.

1.6.5 Динамічне створення документа

У той час як динамічне генерування речень працює на певному "мікрорівні", то динамічне створення документа на більш високому рівні генерує документ, який є релевантним і корисним для його читачів, а також дуже гарно структурований для читача. Процес як це робиться, залежить від мети тексту. Наприклад, якщо на потрібно переконливе написання то структура може базуватися на моделях аргументації та зміні

поведінки, щоб імітувати людську риторику. Інший приклад, текст який підсумовує дані бізнес-аналізу, може базуватися на аналізі ключових факторів, які більше всього впливають на рішення.

1.7 Моделі для NLG

Навіть після того, як NLG перейшла від шаблонізаторів до динамічного генерування, ще багато часу потрібно було витрати щоб знайти оптимальні моделі. Можна відділити певну низку алгоритмів які найчастіше використовується для NLG та мають свої певні сфери використання.

1.7.1 Ланцюги Маркова в NLG

Ланцюги Маркова був одним з перших алгоритмів, використовуваних для генерування мови. Ця модель прогнозує наступне слово у реченні, використовуючи поточне слово та розглядаючи співвідношення між кожним унікальним словом для обчислення ймовірності наступного слова. Наприклад, якщо модель навчалась, використовуючи лише такі пропозиції: "Я п'ю каву вранці" та "Я їм бутерброди з чаєм". То модель буде мати сто процентний шанс на прогноз, що "кава" буде йти "пити", в той час як буде п’ятдесяти процентний шанс, що після "Я" буде йти "пити", а ще п’ятдесят що буде йти "з'їсти". Це доволі відома модель, оскільки вони використовувались у більш ранніх версіях клавіатур для смартфонів для створення рекомендацій наступного слова у реченні.

Однак, зосередившись на поточному слові, моделі Маркова втрачають весь контекст та структуру попередніх слів у реченні, що може призвести до неправильних прогнозів, обмежуючи їх застосовність у багатьох сценаріях генерації мови.

Визначення:

Ланцюг Маркова - це процес X0, X1, X2,. . .XN. Стан ланцюга Маркова в момент часу t - значення Xt . Наприклад, якщо Xt = 6, ми говоримо, що процес знаходиться в стані 6 в момент t. Простір стану ланцюга Маркова, S, - це набір значень кожного Xt може взяти. Наприклад, S = {1, 2, 3, 4, 5, 6, 7}.

Нехай S має розмір N (можливо, нескінченний). Траєкторія ланцюга Маркова - це особливий набір значень для X0, X1, X2,. . .. Наприклад, якщо X0 = 1, X1 = 5 і X2 = 6, то траєкторія до часу t = 2 - 1, 5, 6. Більш загально, якщо ми маємо на увазі траєкторію s0, s1, s2, s3,. . ., маємо на увазі це X0 = s0, X1 = s1, X2 = s2, X3 = s3,. . . "Траєкторія" - це лише слово, що означає "шлях".

1.7.2 Рекурентні нейроні мережі в NLG

Нейронні мережі - це моделі, які намагаються імітувати роботу мозку людини. пропонуючи альтернативний метод обчислення, моделюючи нелінійні зв’язки між вхідними та вихідними даними - їх використання для моделювання мови відоме як Natural language modelling(NLM). Рекурентна нейронна мережа - це тип нейронної мережі, який може використовувати послідовний характер введених даних. Він передає кожен елемент послідовності через мережу й надає вихідну моделі як вхідну для наступного елемента в послідовності, що дозволяє зберігати інформацію з попередніх етапів. «Пам’ять», яку мають рекурентні нейроні мережі, робить їх чудовими для генерації мови, оскільки вони можуть запам'ятовувати контекст речень з яким вони зараз працюють. Рекурентні нейронні мережі відрізняються від ланцюгів Маркова тим, що вони також

використовують слова, які вони раніше «бачили» (на відміну від ланцюгів Маркова, які просто перевіряють тільки попереднє слово), для прогнозування наступного слова.

Приклад роботи рекурентної нейронної мережі:

У кожній ітерації нейронної мережі модель зберігає попередні слова (в пам'яті), що зустрічались, і обчислює ймовірність наступного слова. Наприклад, якщо модель генерувала текст "Нам потрібно взяти напрокат ___", тепер їй потрібно визначити наступне слово у реченні. Для кожного слова у словнику модель призначає ймовірність, беручи свої варіанти з попередніх слів які вона побачила. У нашому прикладі слова "будинок" чи "машина" матимуть більшу ймовірність, ніж, наприклад, слова "річка" чи "вечеря". Слово з найбільшою ймовірністю вибирається і зберігається в пам'яті, а потім модель переходить до наступної ітерації.

Основною проблемою рекурентних нейронних мереж для генерації тексту є проблема зникання градієнту. Зі збільшенням довжини послідовності нейрона мережа не може зберігати слова, які зустрічаються пізно в реченні, і лише робить прогнози на основі останніх слів. Це обмежує застосування рекурентних нейронних мереж для створення довгих речень, які звучать цілісно.

1.7.3 Довга короткочасна пам’ять(ДКЧП) в NLG

У середині 90-х років німецькі дослідники Сепп Хохрейтер та Юрген Шмідхубер запропонували варіацію рекурентної нейронної мережі з деякими одиницями довготривалої пам’яті для вирішення проблеми никання градієнту. ДКЧП допомагають зберегти помилку, яку можна розповсюдити через шари. Підтримуючи більш стабільні помилки, ДКЧП дозволяють рекурентним мережам продовжувати вчитися протягом значно

більшої кількості ітерацій, тим самим відкриваючи канал для зв’язку між причинами та наслідками. ДКЧП мережі містять інформацію за межами нормального потоку рекурсивної мережі в замкненій комірці. Інформація може зберігатися, записуватися або зчитуватися з комірки, подібно до даних у пам'яті комп'ютера. Комірка приймає рішення про те, що зберігати, коли дозволяти читати, писати та видаляти данні, за рахунок «воріт», які відкриваються та зачиняються. На відміну від збереження даних на комп'ютерах, ці ворота є аналоговими, реалізованими з по компонентний добутком на сигмоїди, які знаходяться в межах від нуля до одного. Аналогові дані мають перевагу над цифровими у тому, що вони диференційовані, а тому придатний для зворотного поширення. Ворота діють на сигнали, які вони отримують, і подібно до вузлів нейронної мережі, вони блокують або передають інформацію на основі її важливості, яку вони вираховують за допомогою надання інформації її ваги. Важливість даних, як й ваги які створюють вхідні та приховані стани, регулюються за допомогою процесу навчання мереж під час ітерацій. Тобто, комірки дізнаються, коли дозволити вводити, залишати або видаляти дані за допомогою ітераційного процесу вгадування, розповсюдження, зворотного поширення помилки та коригування ваг за допомогою градієнтного спуску.

Основною перевагою ДКЧП мереж є те що вони дозволяють вибірково відстежувати лише потрібну інформацію, одночасно мінімізуючи проблему зникання градієнта, що дозволяє моделі запам'ятовувати інформацію протягом більш тривалого періоду часу.

Й хоча ДКЧП та його варіації, вирішують багато основні проблеми рекурсивних нейронних мереж, однак й в таких мережах є обмеження щодо скільки інформації можна зберегти, оскільки існує ще складний шлях від попередніх комірок до поточної комірки. Це обмежує довжину

послідовностей, яку ДКЧП мережа може запам'ятати до лише кількох сотен слів. Додатковим недоліком є те, що ДКЧП дуже дорого навчати через високі обчислювальні вимоги. Зважаючи на їх послідовний характер, їх навчати паралельно, що обмежує їх здатність використовувати переваги сучасних обчислювальних пристроїв, наприклад, сучасні графічні процесори.

- Ворота забування (f): визначають, в якій мірі будуть забуті попередні дані.
- Ворота входу (i): вони визначають ступінь інформації, яку потрібно записувати у до комірок.
- Ворота модулювання вхідних даних (g): їх часто розглядають як підрозділ вхідних воріт. Вони використовуються для модуляції інформації, яку вхідний шлюз запише у комірку, додавши не лінійність до інформації та зробивши інформацію нульовою. Це робиться для скорочення часу навчання. Хоча дії цих воріта є менш важливими, є хорошою практикою включати цей єлемент до структури LSTM.
- Вихідні ворота (o): визначають які вихідні дані будуть генеруватись з поточного стану комірки.

  Робота рекурентного блоку ДКЧП:

  1. Взяти вхідні поточні дані, попередній прихований стан і попередній внутрішній стан комірки.
  2. Обчислити значення чотирьох воріт.
  3. Обчислити поточний внутрішній стан комірки, спочатку обчисливши по компонентний добуток вектору вхідних воріт та воріт модуляції даних, потім знайти по компонентний добуток вектору воріт забування та попереднього внутрішнього стану комірки, а потім додати два вектори.

$C_t = i \odot g + f \odot C_t\text{-}1$

4. Обчислити поточний прихований стан, спершу взявши гіперболічний тангенс по компонентно від поточного внутрішнього вектора стану комірки, а потім знайти по компонентний добуток з даними з вихідних воріт.

   $h_t = o \odot tg(C_t)$

1.7.4 Transformer в NLG

Transformer вперше був представлений у документі Google “Attention is all you need” в 2017 році, де було запропоновано новий метод, який називається “self-attention”. В даний час transformer використовуються в широкому спектрі завдань NLP, таких як моделювання мови, машинний переклад та генерування тексту. Transformer складається зі стека енкодерів для обробки вводу будь-якої довільної довжини та іншого стека декодерів для виведення згенерованих речень. Декодер використовує закодовану інформацію для створення вихідного речення слово за словом. Потім модель агрегує інформацію з усіх інших слів, використовуючи “self-attention”, щоб генерувати нові вихідні данні на кожне слово, це дозволяє генерувати контент який враховує контекст повністю. Цей крок повторюється багато разів паралельно для всіх слів, послідовно генеруючи нові данні. Так само, декодер генерує одне слово за іншим, зліва направо, проте він відвідує не лише раніше створені слова, але і остаточне речення, розроблене енкодером.

Одним з найвідоміших прикладів використання «Трансформера» для генерації мови є розроблена компанією OpenAI мовна модель GPT-2. Модель вчиться передбачати наступне слово в реченні, орієнтуючись на слова, які раніше були використанні в моделі та пов'язані за контекстом з наступним словом. Останнє оновлення від Google є презентація

двостороннього енкодера для «Трансформера» який використовується для у великій кількості NLP задач, та генерує одні з найякісніших текстів.

1.8 Використання NLG

- Електронна комерція: NLG використовується у численних інтернет-магазинах для ефективного створення описів товарів повністю автоматично. Таким чином, NLG оптимізує SEO та генерує унікальні описи продуктів у великій різноманітністю текстів та навіть у власному різноманітністю тонів.
- Туризм: ця сфера майже усіляко й повністю залежить від гарних описів, наприклад презентація готелю або курорту. На цьому міжнародному ринку виходить одна з сильних сторін NLG: автоматичний переклад текстів на близький до рівня носія мови.
- Медіа та видавнича діяльність: NLG допомагає видавцям та редакторів в автоматизації створення контенту. Звіти з областей, де проходять великі об’єми даних, таких як спорт, погодні звіти, ситуація щодо руху трафіку та звіти з фондового ринку, можуть створюватися NLG автоматично.
- Енергетика та телекомунікації: для покращення обслуговування клієнтів, NLG можна використовувати, наприклад, для спрощення пояснення великих рахунків у письмовій формі. Також внутрішні дані в мережі допоможуть швидше помітите не точності у підрахуванні оскільки дані записуються автоматично, що допоможе працівниках скоріше помітити помилку.
- Охорона здоров'я: Точне тлумачення даних може бути дуже актуальним у, наприклад, медичному секторі. Тому що дуже важливо щоб медичні висновки а це: діагнози, лікарські рекомендації, лабораторні результати чи результати тестів

медичних пристроїв або ліків, створювались швидко, точно та були легкими для розуміння. NLG може автоматизувати ці процеси і тим самим сприяти підвищенню ефективності.

- Логістика: NLG може інформувати мандрівників або відправника чи одержувача про доставку товару з детальними звітами у текстовій формі у разі виникнення проблем у майбутніх маршрутах мандрівника або затримці товару замовнику на певній стадії маршруту. Це допоможе пришвидшити та покращити контакт між користувачем послуг та виконавцем послуг.
- Фінансові послуги, банківська справа та інвестиції: Банки та фінансові установи в даний час використовують NLG для широкого спектру застосувань, включаючи звітування про фондову ситуацію та рівні акцій, формування кредитних рейтингів, заявки на кредит, генерацію фінансової інформацію, формування бізнес аналітики та інше.
- Страхування: NLG використовується, наприклад, у сфері страхування, наприклад, під час створення звітів про отриману шкоду, або - у поєднанні з NLU використовується в чатах для більш ефективного спілкування з клієнтами в ситуаціях швидкого реагування. Так само як й в інших сферах NLG може автоматизувати створення звітів де корпоративна звітність відіграє важливу роль.
- Продажі, маркетинг та обслуговування клієнтів: Програмне забезпечення NLG вже посилює потужність чат-ботів, що можуть дати більш інтуїтивні відповіді на інколи складні запити клієнтів. Інша сфера застосування - автоматизація пошти в обслуговуванні клієнтів або маркетингу: персоналізовані електронні листи можуть бути автоматично згенеровані та надіслані певним цільовим групам.

1.9 Приклади сервісів NLG

- Text-to-speech сервіси

  Майже всі лідери IT світу мають свої text-to-speech сервіси, а тобто сервіси які генерують текстовий формат в усний формат. Приклади таких сервісів є: IBM Watson Text-to-speech, Amazon Polly, Google Cloud Text-to-Speech та Microsoft Azure Text-to-Speech API. Більшість с цих сервісів підтримують найширше використовуванні мови, як Англійська, Китайська, Іспанська та інші, проте сервіси від Google відрізняються своєю різноманітність мов, включно й Українською.
- Wordsmith

  Wordsmith – платформа що спеціалізується на бізнес аналітиці та звітах даних. Основна функція платформи це генерація даних у читабельні для людини тексти. Wordsmith надає аналіз даних у реальному часі, масштабування використання сервісу в залежності від кількості даних, інтеграція з більшістю інструментів для бізнес аналізу, таких як: Tableau, Power BI, MicroStrategy та TIBCO Spotfire.
- ArticleForge

  ArticleForge – можна вважити контент-райтером для потрібної вам теми. Ви надаєте список ключових фраз які потрібно використати, та тему контенту яку вам треба створити та запускаєте ArticleForge. На відміну від людини для генерації тексту ArticleForge потрібно менше хвилини, а за цей час сервіс зможе підібрати до як медіа файли такі як: відео, зображення, проте й також буде притримуватись SEO щоб ваш контент був максимально оптимізований.

- Ax-semantics

  Ax-semantics – ще одна платформа для генерації тексту з данних. На відміну від Wordsmith ax-semantics визначаються свою експертизу в раніше наведених сферах такі як: медична, страхова та навіть журналістська й інші. Основною перевагою ax-semantics є великий вибір на платформі, як тем, так й мов які можливо використовувати, оскільки платформу обіцяє можливість генерації текстів на 110 мовах світу.

1.10 Що таке Natural Language Understanding(NLU)

Незважаючи на те, що з’явився вже доволі давно, коли програмісти почали експериментувати з природними мовами, NLU почала розвиватися в 60-х роках через бажання змусити комп'ютери розуміти більш складні мовні введення. Як підтема NLP, NLU є вужчою за своїм призначенням, зосереджується насамперед на розумінні машинного читання: змушенні комп’ютера зрозуміти, що насправді означає текст. Хоч NLU я тільки підтипом NLP, більшість людей мають на увазі саме NLU коли говорять про NLP

1.11 Процеси та підходи NLU

Через великий об’єм типів текстових та усних мовних структур для NLU використовується багато підходів та моделей для структуризації та розуміння природної мови.. До недавнього прогресу в моделях deep learning найчастіше використовувані методи були мовні моделі базовані на ймовірностях, такі як n-gram моделі або логарифмічно-лінійні моделі.

Завдання мовної моделі передбачи ймовірність фрагменту текстової послідовності.

1.11.1 Сегментація та токенізація тексту

При лінгвістичному аналізі мовного тексту необхідно чітко визначити, що являє собою слово й речення. Визначення цих підрозділів представляє різні проблеми залежно від мови що обробляється, але жодне завдання не є тривіальним, особливо коли враховуючи різноманітність людських мов та систем письма. Природні мови містять притаманні неоднозначності та більшу частину проблеми природного обробки мови передбачає вирішення цих неточностей.

У сегментація тексту існує завдання поділу тексту на найменші лінгвістичні одиниці - символи найнижчого рівня що представляють окремі графеми в письмовій мови. Сегментація тексту є часто не поміченою, але важливою частиною будь-якої системи обробки природньої мови, оскільки слова та речення, визначені на цьому етапі, є основними одиницями для подальшої обробки на таких стадіях як морфологічний аналіз, розмічування частин мови, парсингу та індексації даних в пошукових системах.

Токенізація - це процес розбиття послідовності символів у тексті , а тобто виділення лімітів де закінчується одне слово, а інше починається. В сфері комп’ютерної лінгвістики такі ідентифіковані слова часто називаються токенами. У письменності, де ліміти слів явно не визначаються , токенізація також відома як морфологічний аналіз, і цей термін часто використовується синонімічно з токенізацією.

Сегментація речень - це процес визначення більших за розміром одиниць що складається з одного або декількох слів. Вона передбачає визначення

меж речення між словами в різних реченнях. Оскільки у більшості писемних мов присутні розділові знаки то часто цей процес називають процесом знаходження меж речення.

На практиці сегментація речень і слів не може бути успішно виконана незалежно одна від одної. Наприклад, важливий під-задача розділових знаків і в слові, і в реченні, наприклад, тому що точка може бути використана як для закінчення речення так й в означенні абревіатури. У випадку коли точка позначає абревіатуру то символ зазвичай вважається частиною лексеми абревіатури, тоді як точка в кінці речення зазвичай вважається окремою лексемою.

1.11.2 Лематизація та стеммінг

При обробці природних мовою дуже значну увагу також треба приділити щоб ваша програма могла розуміти що слова "запитати" і "запитувати" – це просто різні форми одного слова, а не різні слова. В цьому і є ідея зведення різних форм слова до основного кореня. Слова, що походять одне від одного, можна перенести на основне слово або символ, особливо якщо вони мають однакове цільове значення. Це допомагає пришвидшити роботу алгоритму, а також зберегти пам’ять під час роботи алгоритму.

В таких ситуаціях використовуються два підходи, стеммінг та лематизація. В своїй основі два підходить роблять одну річь, проте підходи цих методів відрізняються.

1.11.3 Стеммінг

З двох підходів, безумовно, простішим є стеммінг оскільки в цьому методі слова просто зводяться до їх коренів. Корінь слова під час стеммінгу не обов’язково повинен бути морфологічним коренем слова який зазначений

у словниках, він просто є рівною або меншою формою зазначеного слова. Алгоритми створення зазвичай побудовані на правилах. Ви можете розглядати їх як евристичний алгоритмом який начебто відрізає кінцівки слів. Слово переглядається і проходить через ряд умов, які визначають, як його скоротити.

Однак, оскільки, стеммінг заснована на евристичному алгоритмі, цей підхід є далеко не ідеальним. Й зазвичай стеммінг страждає в двох випадках: overstemming та understemming. Це може призводити до того що якщо відрізати за мало від слова, то його сенс може бути не точним та загубленим. Й навпаки це може призвести до того що слова можуть бути спільнокореневими, проте мати абсолютно різні значення.

Ці проблем можливо швидко побачити при роботі зі складними текстами. Застосування нових правил може створювати нові зіткнення оскільки вирішення однієї або двох проблем що виникають, може призвести до появи ще двох. Скласти хороший алгоритм стеммінгу доволі важко.

1.11.4 Приклади алгоритмів стеммінгу

Porter stemmer:

Цей алгоритм є більш старим. Його використання почалося з 1980-х років, і головним його завданням було усунення закінчень до слів, щоб їх можна було звести до загальної форми. Це не надто складний алгоритм і він майже не розвивася . Це непоганий та простий алгоритм для простих текстів, але його не рекомендується використовувати для будь-якого комерційного застосування або застосування що буде працювати зі складними текстами. На відмінну від інших розроблених алгоритмів porter stemmer не є строгим алгоритмом.

Snowball stemmer:

Цей алгоритм також відомий як алгоритм Porter2 stemmer. Він загально визнано кращий за Porter stemmer, що визнають навіть автори Porter stemmer. Також він є більш строгим при роботі з текстом. Більшість речей які було додано в Snowball stemmer це речі які погано працювали або були відсутні в попередній версії.

Lancaster stemmer:

Ще один алгоритм стеммінгу. Він є найшвидшим з трьох алгоритмів й найсуворішим в обробці тексту. Цей алгоритм доволі часто занадто сильно обрізає корені слова так що вони втрачають свою основну частину майже повністю. Перед використанням рекомендується спочатку протестувати алгоритм щоб побачити як він працює з вашим текстом.

1.11.5 Лематизація

На відміну від стеммінгу, лематизація - це більш обчислений процес. Він передбачає зведення слів до їх словникової форми. Оскільки лематизація являє собою більш складну структуру, то для її використання потребується більше умов. Щоб лематизація зводила слова до їх лем, вона повинна знати до якої частини мови відноситься слово. Для цього потрібна додаткова лінгвістична структура як розмічувач частин мови. Це дозволяє лематизації робити більш якісні рішення на відміну від стеммінгу.

Ще одна річ, яку слід зазначити про лематизацію, полягає в тому, що часто створювати лематизатор новою мовою важче, ніж стеммінг алгоритм. Оскільки лематизаторам потрібно набагато більше знань про структуру мови, а набагато складніший задача ніж спробувати налаштувати евристичні алгоритми як у стеммінгу, тому щоб якісно використовувати лематизацію, потрібно гарно розуміти структуру мови з якою будеш працювати або використовувати вже існуючи варіанти.

1.11.6 Маркування частин мов

Обробка мовних даних також потребує у деяких випадках розуміти до якої частини мови відноситься слово. Розмічування частин мови – це процес відношення слова у вхідних даних до потрібної частини мови, як згадувалося раніше, розмічування частин мови є обов’язковим для використання лематизації. Цей процес можна описати як маркування слова в корпусі відповідної частини мови, маркування відбувається за рахунок контексту та значенню слова. Це завдання не є простим, оскільки слово може мати бути різною частиною мови в залежності від контексту, в якому слово використовується.

Існують різні методи для розмічення частин мови

- Лексичні методи – Маркує слово певною частиною мови в залежності від частоти використання слова в навчальних корпусах.
- Методи засновані на правилах – маркують слово на основі правил. Наприклад, у нас може бути правило, яке говорить, що слова, що закінчуються на "ться" або “шся”є дієсловами. Методи засновані на правилах, можуть використовуватися разом з лессичними методами, що дозволить маркувати слова яких немає в начальних корпусах, проте вони присутній в тестових даних.
- Методи засновані на ймовірності – цей метод маркує слово до певної частини мови на основі ймовірності певної послідовності маркувань. Умовні випадкові поля та Приховані Моделі Маркова є основами для цих методів.
- Методи на основі нейронних – Алгоритми рекурентних нейронних мереж також використовуються для маркування частин мови.

1.11.7 Regexp

Регулярні вирази, або Regexp - один з найбільш використовуваних, у для стандартизації та пошуку патернів тексту в галузі інформатики Цей інструмент використовується майже в усіх мовах програмування та інструментах в операційних системах, таких як grep в UNIX. Формально регулярний вираз є алгебраїчне позначення характеристики набору рядків. Регулярні вирази особливо корисні для пошуку текстів, коли у нас є шаблон пошуку та корпус тексту для пошуку. Функція пошуку регулярних виразів здійснюватиме пошук за допомогою корпусу тексту, повертаючи весь текст що відповідають зразку. Корпусом може бути окремий документ або колекція. Наприклад, grep командного рядка Unix приймає регулярний вираз і повертає кожен рядок вхідного документа, що відповідає виразу. Пошук може бути розроблений таким чином, щоб повертати кожну відповідність у рядку, якщо їх більше ніж одна, або можна повертати тільки перше співпадіння. Ви можете легко витягувати, замінювати та робити різні операції з маніпуляції з рядками, використовуючи регулярні вирази. Регулярні вирази самі по собі є мовою, оскільки вони мають власні компілятори. Використання тестерів регулярних виразів в Інтернеті - це зручний спосіб перевірити свої вирази. Не буде перебільшенням вказати, що не розуміючи регулярних виразів, неможливо реально побудувати систему на базі NLP, таку як чат-боти, розмовні інтерфейси, тощо.

1.11.8 N-gram моделі

N-gram модель є типом імовірнісної мовної моделі для прогнозування наступного елемента в такій послідовності у вигляді (n - 1) за моделью ланцюгів Маркова.

Основною задачею n-gram моделі є знаходження вірогідності слова w при історії h P(w|h). Один зі способів підрахувати вірогідність це підібрати величезний корпус та порахувати кількість послідовних зустрічей слова w з іншим словом. Проте знайти занадто великий корпус для підрахунку майже неможливо оскільки постійно достатньо гнучка та варіації речень можуть в дуже великій кількості. З цієї причини потрібен знадобиться ввести кращий спосіб для підрахунку ймовірності слова w з історією h або ймовірності всієї послідовності слів W. Ми можемо представити послідовність слів N як w_1^n. Використовуючи ланцюгове правило ми отримаємо.

$$\begin{aligned} P(w_1^n) &= P(w_1)P(w_2|w_1)P(w_3|w_1^2)\dots P(w_n|w_1^{n-1}) \\ &= \prod_{k=1}^{n} P(w_k|w_1^{k-1}) \end{aligned}$$

Сенс n-грамової моделі полягає в тому, щоб замість обчислення ймовірності слова, враховуючи всю його історію, ми можемо вирахувати приблизну історію лише за останні кілька слів.

Біаграмна модель, наприклад, обирає наближене значення використовуючи всі попередні слова: $P(w_n|w_1^{n-1}$¿ використовуючи тільки умовну вірогідність попереднього слова $P(w_n|w_{n-1}$¿ . Припущення, що ймовірність слова залежить лише від попереднього слова називають припущенням Маркова.

Щоб порахувати точність n-gram моделей ми можемо використати метод максимальної правдоподібності.

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{\sum_w C(w_{n-1}w)}$$

1.11.9 Наївний баєсів класифікатор

Naive Bayes - імовірнісний класифікатор, що означає, що для документа d, для всіх класів c ∈ C класифікатор повертає клас ˆc, який має максимум апостеріорної ймовірність документа.

Правило Байєса дає нам можливість розбити будь-яку умовну ймовірність P (x | y) на три інші ймовірності:

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)}$$

Наївний баєсів класифікатор робить два припущення:

1. Перший – це «торба слів» ми припускаємо що позиція не має значення, і те, що слово має однаковий вплив на класифікацію чи зустрічається вона як перше чи останнє слово в документі. Таким чином, ми припускаємо що функції f1, f2, ..., fn кодують лише ідентичність слова, а не позицію.

2. Друге, називають наївним припущенням Байєса: це те що умовні вірогідності P(fi|c) є незалежні даними клас c, а отже їх можна помножити так:

$$P(f_1, f_2, ..., f_n|c) = P(f_1|c) \cdot P(f_2|c) \cdot ... \cdot P(f_n|c)$$

Фінальне рівняння наївного Баєсовського класифікатора є:

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{f \in F} P(f|c)$$

1.11.10 Нейронні мережі прямого поширення

Нейроні мережі прямого поширення є багатошарові перцептрони в яких «нейрони» з’єднані без циклів, а тобто вихідні данні передаються тільки на наступні шари й ніколи на попередні.

- W: Вага з'єднання
- I: вузол введення
- H: Прихований вузол
- HA: Прихований вузол активовано
- O: Вузол виведення
- OA: вихідний вузол активований
- B: Вузол зміщення
- E: Загальна різниця між виходом мережі та бажаним значенням

Спочатку потрібно всі початкові ваги та встановити вузол зміщення.

Після отримування вхідних даних рахуються данні на нодах прихованого шару.

$$H_1 = I_1W_1 + I_2W_3 + B_1W_5$$
$$H_2 = I_1W_2 + I_2W_4 + B_1W_6$$

Далі обраховуються активовані дані прихованих нод з обраною функцією активації, наприклад сигмоїдою:

$$HA_1 = \frac{1}{1+e^{-H_1}}$$
$$HA_2 = \frac{1}{1+e^{-H_2}}$$

Обраховуються дані нод шару виведення

$$O_1 = HA_1W_7 + HA_2W_9 + B_2W_{11}$$
$$O_2 = HA_1W_8 + HA_2W_{10} + B_2W_{12}$$

Вираховуються активовані дані нод виведення, з обраною функцією активації, наприклад лінійною функцією:

$$OA_1 = O_1$$
$$OA_2 = O_2$$

Підрахунок помилки:

$$e = \frac{1}{n}\sum_{i=1}^{2}(y_i - OA_i)^2$$

1.11.11 Word2Vec, Glove

Однією з головних проблем мовного аналізу є метод перетворення тексту на числовий ввід, що робить моделювання можливим. У завданнях, наприклад, computer vision це не є проблемою через те, що на зображенні кожен піксель представлений трьома числами, що зображують насиченість трьох основних кольорів. Так протягом багатьох років дослідники намагалися численні алгоритми для пошуку так званих вкладень, які відносять до подання тексту в виді вектору.

Спочатку більшість цих методів ґрунтувалися на підрахунку слів або коротких послідовностей слів (n-грамів). Початковий підхід до вирішення цієї проблеми - це унітарне кодування, де кожне слово з наданого словника представлене як унікальний бінарний вектор з лише одним ненульовим значення. Основним недоліком цього методу є дуже великий розмір, кожен вектор має за розміром такий самий як й наданий словник (або навіть більший у разі n-грамів), що ускладнює моделювання. Ще один недолік цього підходу - відсутність семантичної інформації. Це означає, що всі вектори, що представляють окремі слова, рівновіддалені. У цьому методі синоніми настільки ж рівновіддалені один від одного як і абсолютно неспоріднені слова. Використання такого методу моделювання

створює зайве завдання, оскільки воно змушує вашу модель запам’ятовувати певні слова, а не намагатися зафіксувати семантику.

Перший великий стрибок у цій галузі відбувся у 2013 році, коли було введено Word2Vec - модель нейронної мережі, що використовується виключно для моделювання векторного уявлення слова(embedding). Модель має змогу розуміти семантику речення, навіть якщо ми дамо їй послідовність слів, а потім заберемо слова посередині, модель все одно зможе зрозуміти семантику тільки за контекстними словами. Це було можливо робити в Word2Vec за допомогою алгоритму Continuous Bag of Words(CBOW). Зворотній підхід розуміння семантики речення за середніми словами, та відсутніми початковими та кінцевими словами пропонував алгоритм Skip-Gram.

У 2014 році, дослідницька група Стенфорда розробила: GloVe. Вони запропонували інший підхід, стверджуючи, що найкращий спосіб кодування семантичного значення слів у векторах - це глобальна матриця збігів слів, на відміну від локальних збігів слів, як у Word2Vec. Відношення ймовірностей збігів здатне розрізняти цільові слова в порівнянні з контекстним словом. Наближене до одиниці чи один ймовірність дорівнює коли обидва цільові слова спільно зустрічаються дуже часто або дуже рідко з контекстним словом. Тільки тоді, коли контекстне слово спільно зустрічається з одним із цільових слів, співвідношення є дуже маленьким або дуже великим. Точний алгоритм передбачає подання слів як векторів таким чином, що їх різниця, помножена на контекстне слово, дорівнює відношенню ймовірностей спільного виникнення.

Виникнення цих двох моделей нейронних мереж сильно покращило використання вже пригаданих способів обробки мови базованих на

нейронних мережах, такий як ДКЧП та звичайні рекурентні нейронні мережі.

1.12 Приклади програмного забезпечення з NLU

- Машинний переклад

  Більшість машинних перекладів на сьогодні зроблені з використовуванням нейронних мереж. Приклади цього програмного забезпечення можна побачити від Google, яке й є найпопулярнішим, проте машинний переклад вже можна побачити й у інших гігантів IT: Microsoft, IBM, Facebook.
- Розпізнавання мовлення

  Розпізнавання мовлення зараз є поширеним як ніколи, популярність можна пояснити масовим використанням голосових помічників. Google Assistant – у Google, Cortana – Microsoft, Alexa – Amazon, Siri – Apple.
- Аналіз емоцій

  Аналіз емоцій є ще не на стільки популярним, проте розвиток можна побачити тому що інструменти надають такі компанії як: H2O.ai – зі своєю платформою Driverless.ai, Scale.ai, Lexalytics. Використання аналізу емоцій можна побачити в Grammarly, вони намагаються надати оцінку емоційного забарвлення написаного користувачем тексту.
- Чат-боти

  Чат-боти набрали неймовірну популярність й хоча не всі з них використовують NLP, все більша кількість ботів починають використовувати NLP. Пригадуючи відомих чат-ботів з NLP можна назвати: X.ai, Xiaoice, Mitsuku.

2. Чат-боти

2.1 Що таке чат-бот

Chatbot можна визначити як комп'ютерну програму на яка імітує людську розмову. Також можна визначати як цифрового помічника який створений допомагати людям у різних питаннях. Боти обробляють запити користувачів та дають швидкі та правильні відповіді. Ботами також можна керувати за допомогою голосу, а використовувати їх можна на більшості платформ з можливостю текстового спілкування, наприклад: Facebook Messenger, Whatsapp або Telegram .

2.2 Як працюють чат-боти

Чат-боти працюють шляхом аналізу та виявлення наміру запиту користувача за допомогою вилучення відповідних потрібних для бота об'єктів, це і є найважливішим завданням чат-бота. Після проведення аналізу запиту користувача, йому приходить відповідь за запитом який користувач відправив. Чат-боти працюють, застосовуючи три методи класифікації.

2.2.1 Чат-бота побудований на бізнес правилах

Це найпростіший тип чат-ботів на сьогодні й можливо найпопулярніший. Люди взаємодіють із цими ботами, натискаючи кнопки та використовуючи попередньо визначені команди. Щоб дати відповіді на запити користувача, ці чат-боти вимагають від користувача використання вже існуючих команд. Як результат, ці боти більш складні в використанні для

середнього користувача. Ці боти чудові для класифікації ваших клієнтів, тому що чат-бот задає питання які визначав розробник, а клієнт може відповідати тільки з відповідями які йому пропонують. Також такі чат-боти часто використовуються більше як інструмент для певних задач в чатах: модерування, голосування та інше.

2.2.2 Чат-боти з NLU

У чат-ботах з NLU:

- Суб'єкт: Це сама ідея, а тобто сфера використання чат-бота. Наприклад система оплати, або розклад
- Контекст: Коли алгоритм NLU мов вивчає речення, він не має історії текстової розмови користувача. Отже, фази під час розмови чату зберігаються окремо. Це можуть бути банери типу "Замовлення піци". Або може включати інші параметри, наприклад "Піццерія Домінос". За контекстом можна легко зрозуміти очікування клієнта.
- Очікування: Це те що повинен зробити бот коли після отримання запиту користувача. Очікування може бути однаковим навіть при різних контекстах. Наприклад, поставлена мета: "Я хочу придбати білу пару взуття" та "Чи є у вас білі черевики? ", "Я хочу їх придбати "або" покажіть мені білу пару взуття ",очікування є однаковим продаж взуття.

2.2.3 Чат-боти з NLP

Чат-боти з NLP для обробки мови знаходять спосіб перетворює текстові або усне мовлення користувача у структуровані дані. Щоб потім

використовується для вибору потрібної відповіді. NLP в чат-ботах включаэ наступні єтапи:

1. Токенізація: NLP відокремлює ряд слів на лексеми або фрагменти, які є лінгвістично репрезентативними та з різним значенням .
2. Нормалізація: Модель програми обробляє текст, щоб з’ясувати типографічні помилки та поширені орфографічні помилки, які можуть змінити зміст запиту користувача.
3. Розпізнавання іменованої особи: Програма моделі чат-боту шукає різні категорії слів, подібні до найменування конкретного продукту, адреси або імені користувача, залежно від необхідної інформації.
4. Розбір залежностей: чат-бот здійснює пошук теми, дієслів, об'єктів, загальних фраз та іменників у тексті користувача, щоб виявити споріднені фрази, які користувачі хоче передати.

3. Вибір інструментів розробки

3.1 Які інструменти потрібні

Для розробки чат-бота з елементами NLP потрібен сервіс чи фреймворк які би дозволяли розрізняти Українську мову, та переводив її в формат структурованих даних, також потрібен інструмент для розробки чат-боту для мессенджеру Telegram, або сервіс який поєдную обидві характеристики

3.2 Amazon Lex

Amazon Lex - сервіс для створення голосових і текстових діалогових інтерфейсів в будь-яких додатках. Amazon Lex надає розширені функціональні можливості глибокого навчання, такі як автоматичне

розпізнавання мови (ASR), призначене для перетворення мови в текст, і розуміння природних мов (NLU), призначене для визначення сенсу тексту.

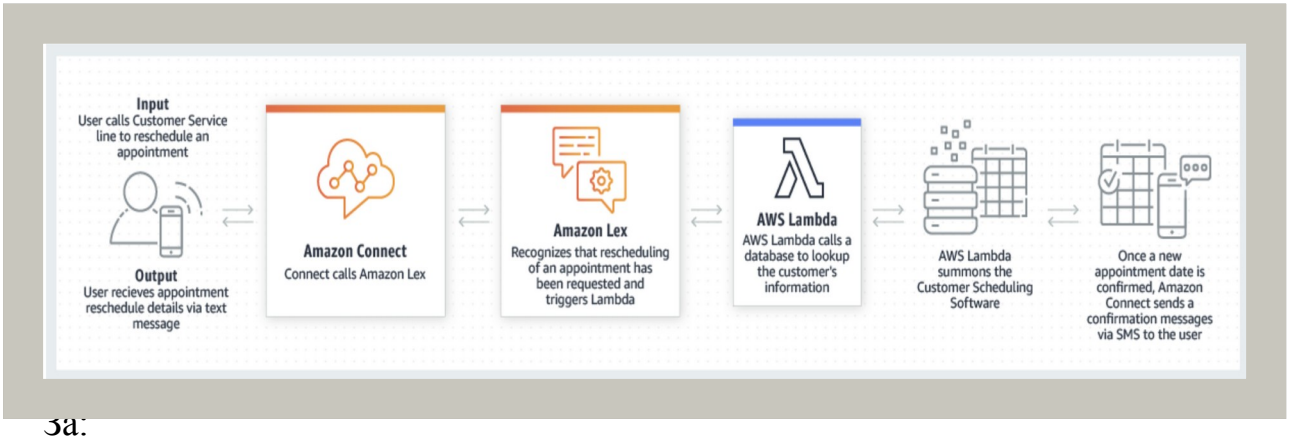
- Інтеграція

  Інтеграція Amazon Lex є тільки в платформи: Facebook, Kik, Slack, Twilio SMS
- Мови

  Amazon Lex підтримує тільки Американську Англійську мову
- Легкість використання

  Amazon Lex має зрозумілий для використання інтерфейс, та два готових приклади чат-ботів дуже допомагають у розумінні процесу розробки. Швидке тестування за рахунок вбудованого веб-чату.

Приклад архітектури з використанням сервісу Amazon Lex.



За:

- Швидкість розробки, Amazon Lex є найшвидшим для розуміння процесу використання.
- Amazon є найпопулярнішою хмарною платформою на сьогодні, тому рівень довіри до самої платформи вищий

Проти:

- Відсутність Української мови
- Відсутність інтеграції в Telegram

3.3 IBM Watson Assistant

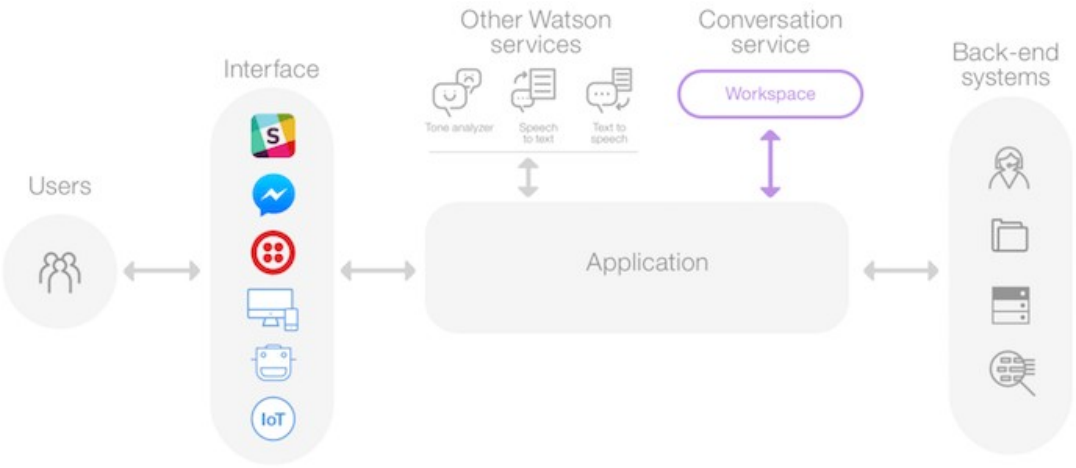
Watson Assistant - це AI-продукт IBM, який дозволяє створювати, навчати та розгортати розмовні інтерфейси в будь-яку программу, пристрії чи канал. Більшість чат-ботів намагаються імітувати людські взаємодії, що може розчарувати користувачів, коли виникає непорозуміння. Watson Assistant знаю коли шукати відповідь у базі знань, коли переспитатти та коли спрямовувати користувача до людини. Watson Assistant може бути розгорнутий у будь-якому хмарній платформі чи локальному середовищі.

- Інтеграція:

  Watson Assistant можна інтегрувати до Facebook Messenger, Slack, плагіну WordPress використовувати в телефонії, а також за рахунок API приєднати до будь-якого додатку користувача.

- Мови
  IBM Watson підтримує більше 10 мов, проте велика частина з них ще на стадії Beta.
- Легкість використання
  IBM Watson пропонує дуже комфортний для використання інтерфейс. Вбудований веб-чат для тестування. Комфортна документація та великий вибір прикладів та гайдів.

Приклад архітектури з використанням сервісу Amazon Lex.



За:

- Можливість деплою свого бота будь де, та можливість інтеграції в будь який додаток
- Більш розумний підхід у вирішенні задач коли чат-бот не розуміє що від нього вимагає користувач.

Проти:

- Відсутність Української мови
- Хмарні платформи від Google, Microsoft та Amazon краще будуть працювати зі своїми вбудованими сервісами ніж с зовнішнім сервісом.

3.4 Azure Bot Framework

Від базових чатів до розумних віртуальних помічників, ви можете створити майже все, що стосується Microsoft Bot Framework. Microsoft пропонує систему розробки чат-бот та суміжні сервіси, що дозволяють розробку чат-бота з великим обсягом функціоналу. Й хоча BotFramework сам по собі не включає NLP, Azure пропонують когнітивні сервіси які можна приєднати до чат-бота. До цих сервісів входить LUIS який якраз дозволяє приєднати сервіс NLP.

- Інтеграція:

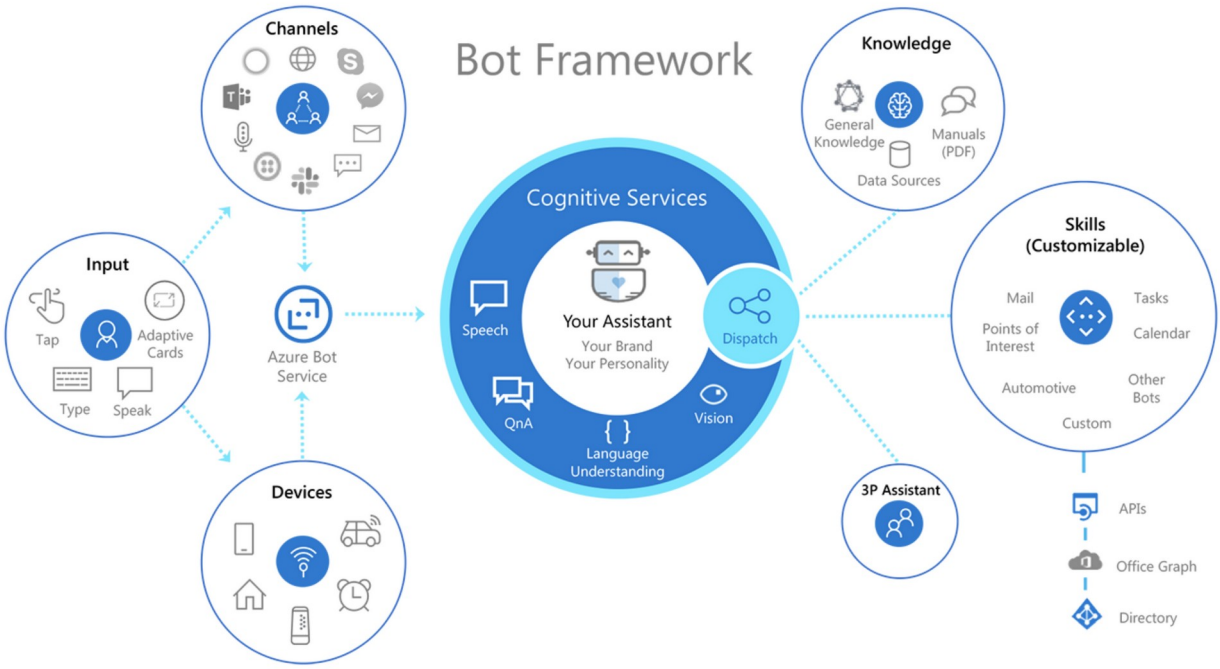
Bot Framework можна приєднати до: Facebook, Messenger, Kik, Skype, Slack, Microsoft Teams, Telegram, SMS, Twilio, Cortana та Skype.

- Мови

  Luis розпізнає більше 10 світових мов на різному рівні розуміння.

- Легкість використання
  BotFramework пропонує SDK на 3 мовах програмування. Для тестування існує окрема програма BotFrameworkEmulator або у вбудованому веб-чаті. Тестування в емуляторі дозволяє гарно розділити проекти, тому її комфортніше використовувати коли ти розробляєш декілька чат-ботів одночасно.

Приклад архітектури з використанням сервісів від Azure



За:

- Можливість використовувати великий обсяг сервісів від Azure разом з чат-ботом
- Комфортна середа розробки

Проти:

- Відсутність Українською на пряму в LUIS
- Не найкомфортніша інтеграція в різні сервіси

3.5 Google DialogFlow

Google теж не ігнорував тренд розробки чат-ботів та запропонував своє рішення. DialogFlow включає всі функції, які ви очікуєте від повноцінного фреймворку для розробки чат-бота. Можливості NLP для розуміння намірів користувачів. Машинне навчання, яке навчає вашого бота для отримання все більш цінних відповідей в залежності від його взаємодії з користувачами. Масштабованість за допомогою хмарних рішень Google. Інтеграція в найрізноманітніші мессенджери та соціальні мережі.

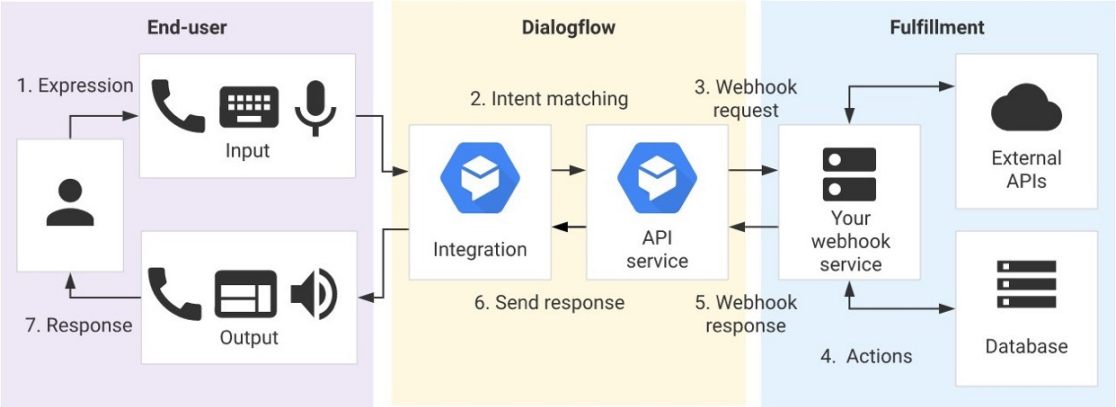
- Інтеграція:

  Інтеграція DialogFlow може відбуватися як у кастомні чат рішення, так й у більшість з існуючих мессенджерів.
- Мови

  DialogFlow підтримує більше 20 мов на різному рівні, включаючи Українську.
- Легкість використання

  Вбудований чат для тестування, проте тестувати можна на пряму в потрібному мессенджері. Не найкомфортніша документація, проте великий обсяг відео на тему. Готові агенти для чат-бота які є чудовими прикладами.

Приклад архітектури з використанням DialogFlow



За:

- Швидка інтеграція в більшість платформ
- За потреби до DialogFlow можливо з’єднатись по API на більшості мов програмувань які зараз використовується для розробки серверної частини.
- Підтримка Української мови та більш стабільна підтримка інших мов, а не тільки Англійської.

Проти:

- Документація перший час є не найкомфортнішою

3.6 Вибір

До фінального вибору було обрано два рішення від Microsoft Azure та Google. Оскільки два рішення можна було з’єднати з Telegram та обидва рішення можна було використовувати з Українською(DialogFlow на пряму, BotFramework з приєднання двом когнітивних сервісів). Проте вибір впав на DialogFlow оскільки використання декількох шарів обробки тексту у BotFramework, а тобто переклад з Української на Англійську і тільки потім обробка за допомогою Luis дуже знижувала точність

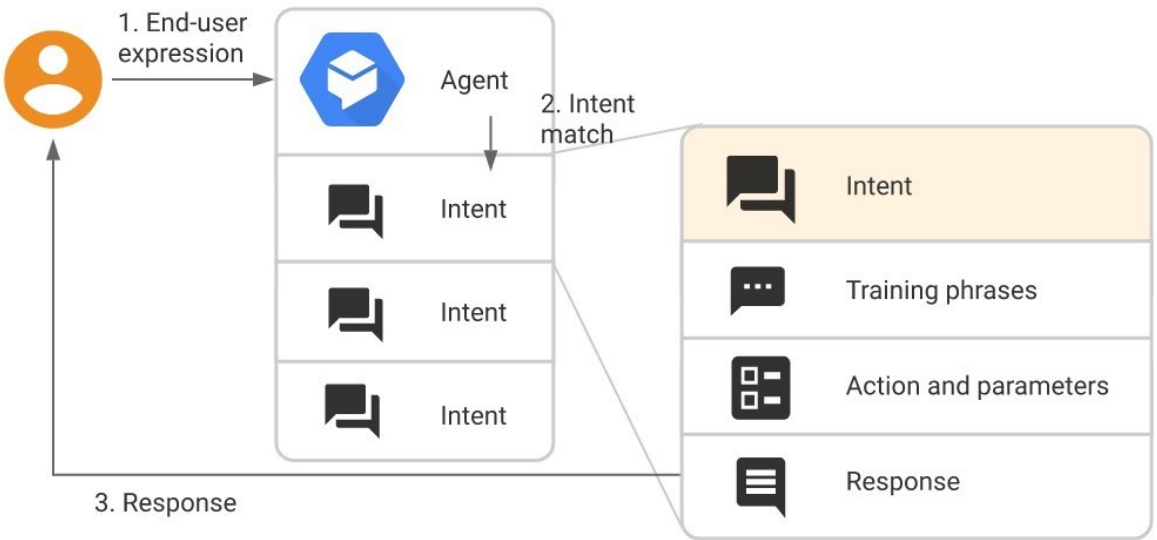
відповідей, та не є дуже економним процесом для додатку оскільки ми робимо зайві операції.

4. Практична Частина

4.1 Загальний опис задачі

Загальне завдання мого проекту була побудова чат-боту для абітурієнтів Факультету Інформатики за для більш швидкого ознайомлення з потрібною інформацією. Під час підготовки контенту для використання у чат-боті можна було побачити що не дивлячись на те що інформацію для вступу, загальних питань та інформація по Факультету Інформатики присутня, проте вона знаходиться на різних платформах(Сайти та Телеграм Канали). Тому задачею було створити програму де абітурієнт міг би знайти та отримати доступ до потрібної інформації в одному місці.

4.2 Архітектура роботи DialogFlow



Як ми можемо побачити, архітектура роботи складається з таких блоків:

- Agent – найвищий блок
- Intent – будівні блоки агента
- Training phrases – фрази для тренування Intent
- Action and parameters – Дії та параметри який отримує та робить Intent
- Response – Фінальна відповідь до користувача

4.3 Agent – Агент

Агент Dialogflow - це віртуальний агент, який обробляє розмови з кінцевими користувачами. Це модуль який розуміє нюанси людської мови. Dialogflow переводить текст або аудіо користувача під час розмови в структуровані дані, які можна використовувати в програмах та сервісах розробників. Агент Dialogflow подібний до агента телефонного центру. Ви навчаєте їх обробляти очікувані сценарії розмови, проте ваше навчання не повинно бути слово в слово.

4.4 Intent – Намір

Intent категоризує цілі користувача на один з напрямів розмов людини. Для кожного агента ви визначаєте багато Намірі, де вони можуть обробляти повну розмову з користувачем. Коли кінцевий користувач щось пише або говорить, Dialogflow намагається підібрати найкращий Намір для відповіді користувачу. Підбір Намірів також відома як класифікація намірів.

Наприклад, ви можете створити агента для погоди, який розпізнає і відповідає на запитання користувачів про погоду. Вам потрібно буде

визначити наміри для запитань щодо прогнозу погоди. Якщо кінцевий користувач каже "Який прогноз?", Діалоговий потік відповідатиме цьому виразу кінцевого користувача цільовим прогнозом. Ви також можете визначити свої наміри щоб отримати корисну інформацію з виразу кінцевого користувача, наприклад, час або місцеположення для бажаного прогнозу погоди. Ці витягнуті дані важливі для вашої системи для виконання запиту що до погоди для точної відповіді користувачу.

4.5 Training Phrases – навчальні фрази

Навчальні фрази - це приклади фраз які можуть вводити або говорити кінцеві користувачі. Для кожного наміру ви створюєте навчальні фраз. Коли один з виразів користувача нагадує одну з цих фраз, DialogFlow відповідає користувачу відповідним наміром. Наприклад, навчальна фраза "Я хочу піцу" навчає вашого агента розпізнавати вирази користувачів, подібні до цієї фрази, як-от "Отримати піцу" або "Замовити піцу". Не потрібно визначати кожен можливий приклад, оскільки вбудоване машинне навчання Dialogflow розширюється у вашому списку іншими, подібними фразами.

4.6 Actions and parameters – дії та параметри

Дія - це просте поле, яке допомагає виконувати певну логіку у вашій програмі. Створюючи агент, ви можете встановити це дію на будь-який текст, який вам здається корисним. Коли намір виконується, DialogFlow виконує визначену вами дію.

Параметри – можна сприймати як звичайний параметр для функції. Наприклад, коли ми шукаємо погоду, ми вказуємо місто погода якого нас цікавить, це і вважається параметром. Параметри можуть бути обов’язкові, або не обов’язкові.

4.7 Responses – Відповіді

Відповідь – назва говорить сама за себе, це відповідь користувача на його запит, це може бути або визначена відповідь, або відповідь яку клієнт отримує згідно параметрів які він передав.

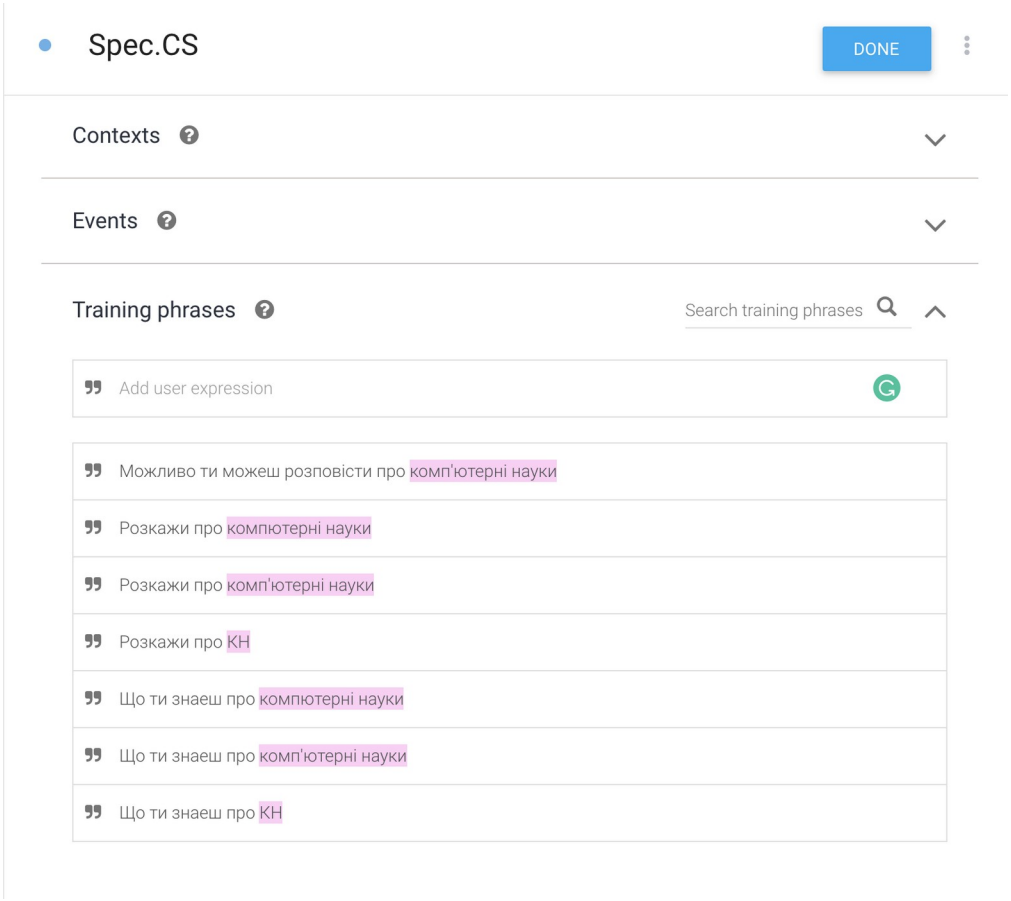
4.8 Entity – об’єкти

Entity - ще один важливий компонент DialogFlow це Entity. DialogFlow має попередньо визначені системні об'єкти, які можуть відповідати багатьом загальним типам даних. Наприклад, є системні об'єкти для відповідності дати, часу, кольорів, адрес електронної пошти тощо. Ви також можете створити власні спеціалізовані об'єкти для відповідності користувацьким даним. Наприклад, можна визначити Факультети, або програму навчання яку хочу передати користувач.

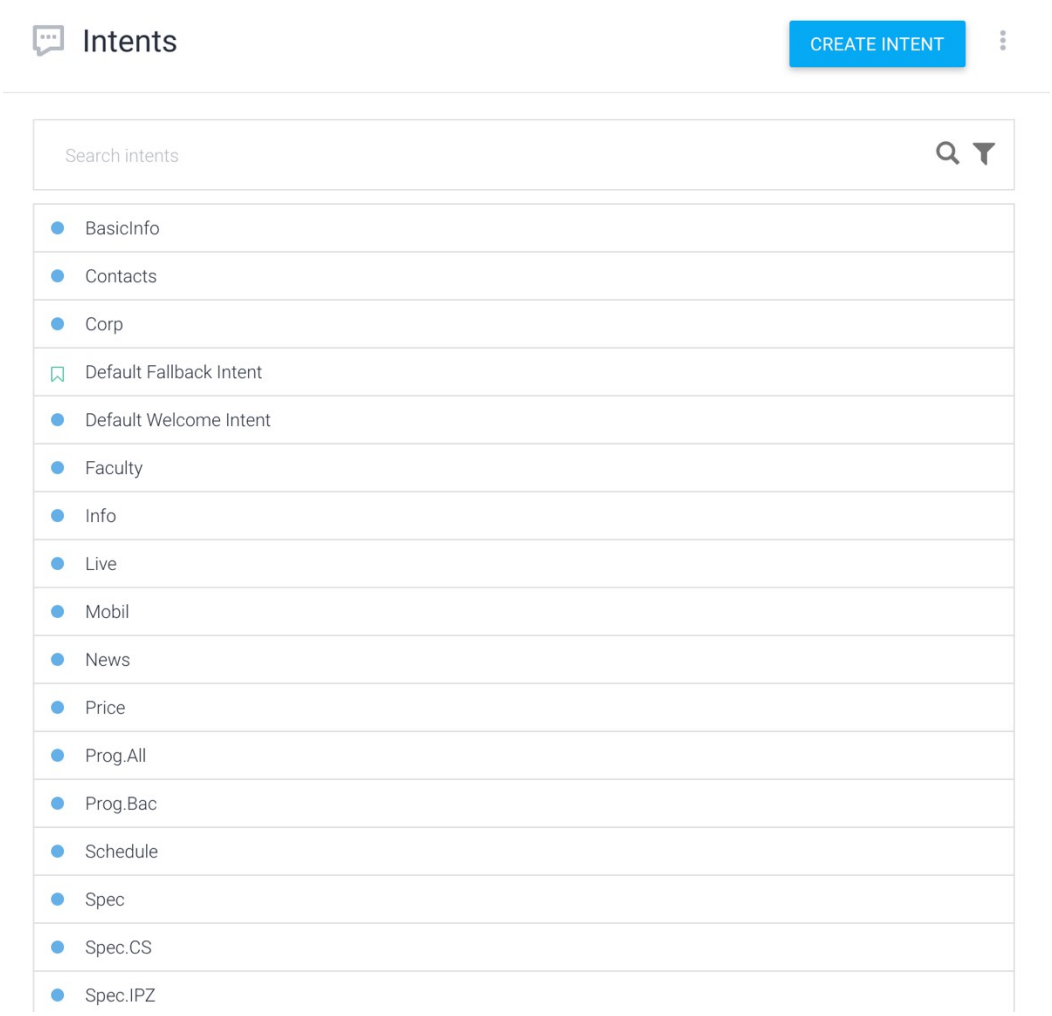
4.9 Використання намірів

Після побудування верхнього рівня, а тобто агенту далі йшли наміри.

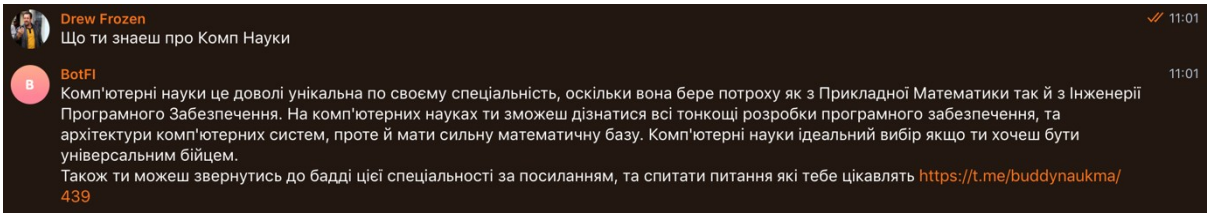
Приклад одного з намірів



Набір частини використаних намірів:

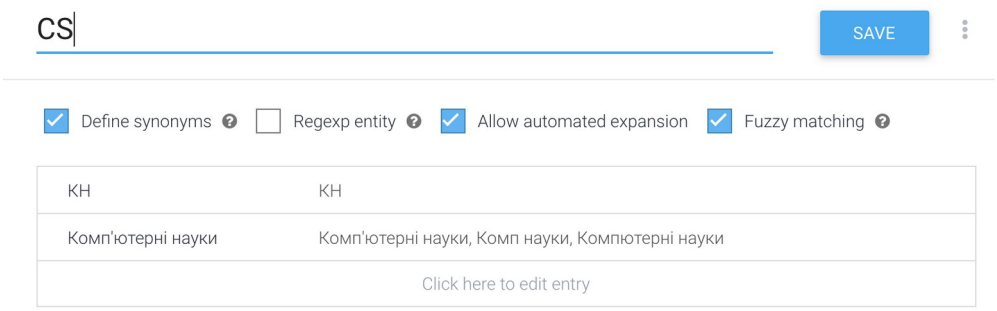


Приклади роботи наміру:

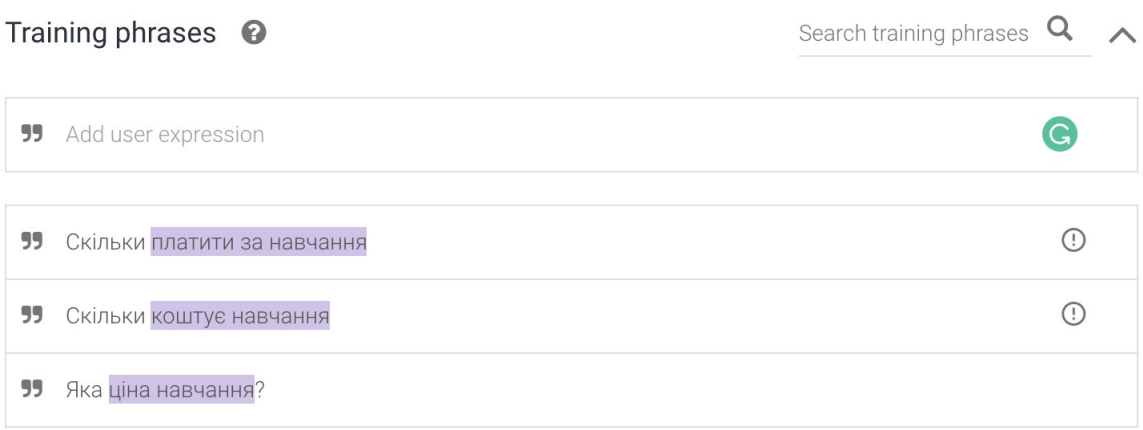


4.10 Використання параметрів та об’єктів

Приклад об’єкту “CS”:



Намір отримання ціни навчання:



Приклади параметрів які передаються на дію:

Prompts for "spec"

NAME	ENTITY	VALUE
spec	@SpecDB	$spec

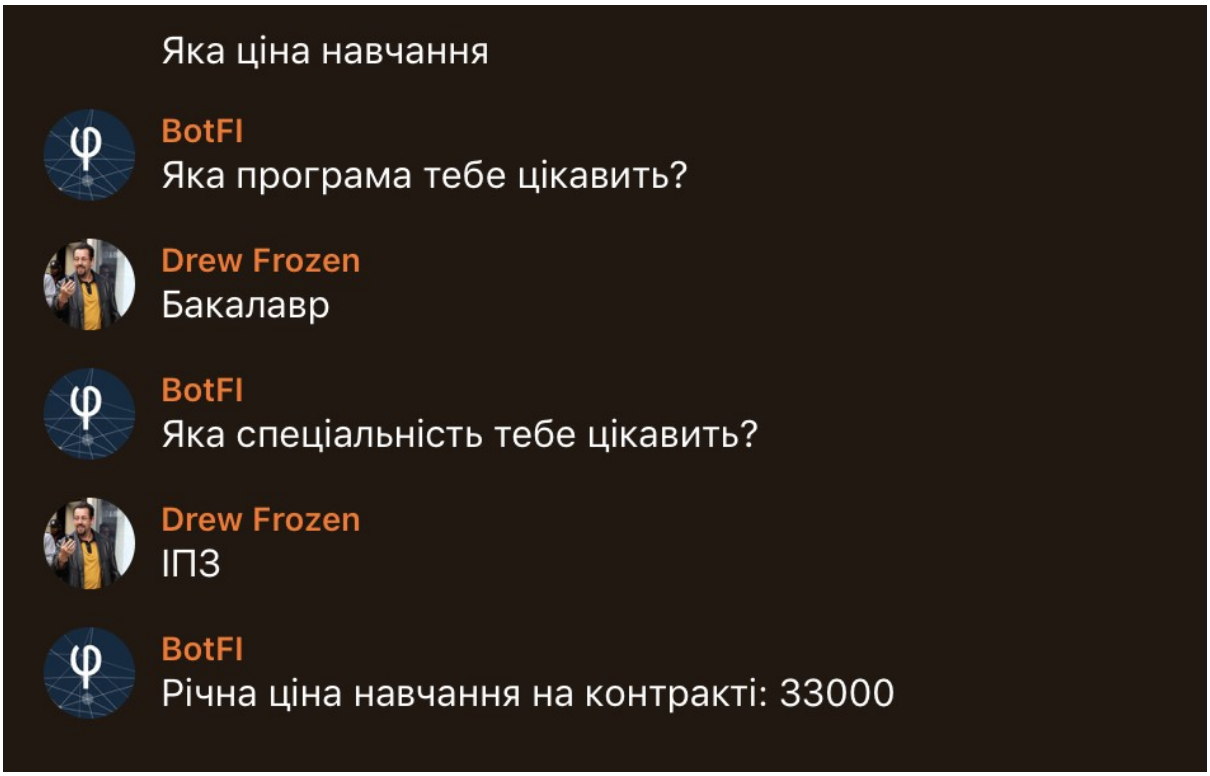
	PROMPTS
1	Яка спеціальність тебе цікавить?

Prompts for "prog"

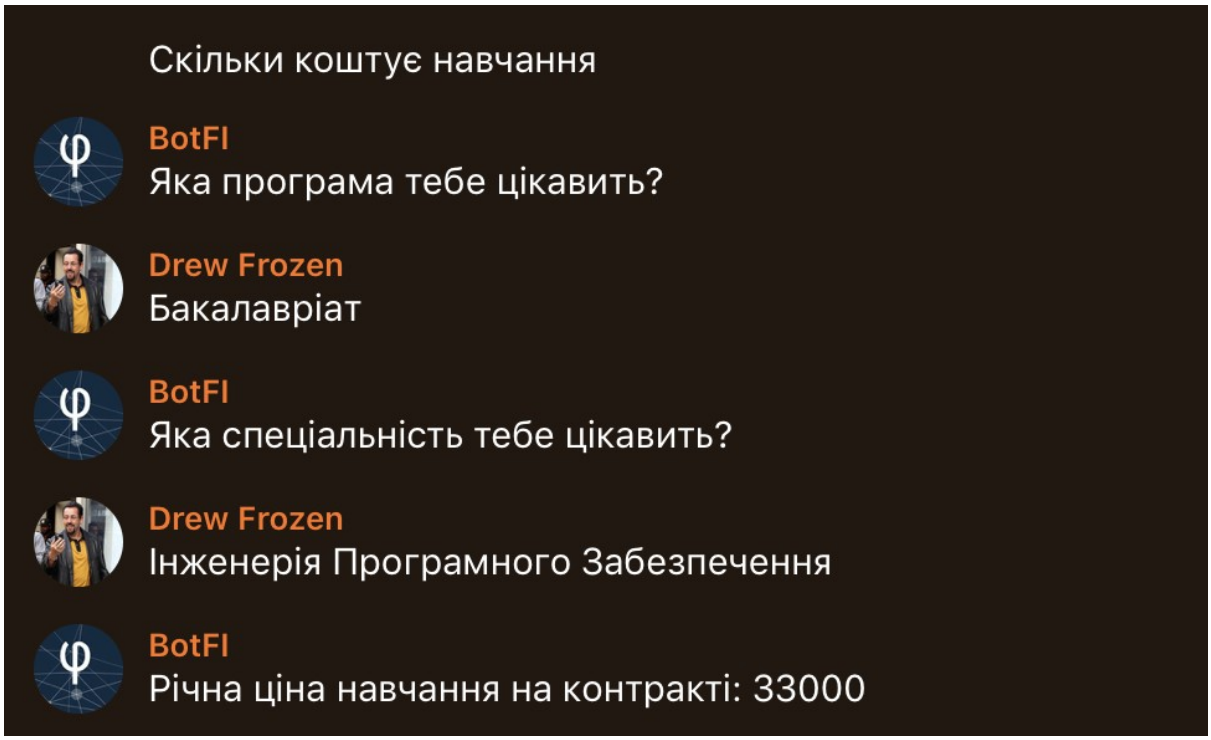
NAME	ENTITY	VALUE
prog	@ProgDB	$prog

	PROMPTS
1	Яка програма тебе цікавить?

Приклади використання параметрів при роботі з Базою данних:



За рахунок використання об’єктів нам не важливо в якій формі ми називаємо спеціальність або программу навчання:



Також за рахунок того щи можемо передавати декілька параметрів ми можемо розділяти однакові за назвою об’єкти.

4.11 Використана база даних

Оскільки база даних не використовується занадто активно у проекті т й в проект не має залежності між даними то ER модель до Бази даних відсутня.

Використана база даних MySQL є :

- Захищеною
- Швидкою
- Проста у використанні
- Можна адмініструвати великою кількістю інструментів

Загалом вибір бази даних скоріше залежав від особистої преференції.

4.12 Хостинг

Оскільки DialogFlow є системою яка працюю виключно на Google Cloud то чат-бот й було розміщено на цьому хмарному сервісі. База даних спочатку була теж розміщена в Google Cloud, проте потів була перенесена на особистий сервер, за для економії ресурсів.

Висновки

Під час роботи над цією курсової я зміг ознайомитись з поняттями Natural Language Processing. Зрозумів різницю між його Natural Language Generation та Natural Language Understanding. Зрозумів методи та стадії готування тестової та усного мовлення для обробки та переведення в структуровані дані. Розібрав алгоритми навчання на розуміння мовлення людини. Розглянув типи чат-ботів які можуть бути створенні та ознайомився з популярними інструментами які дозволяють розробляти чат-ботів з можливостями NLP. Зібрав інформацію потрібну для

абітурієнта в одному місяці, та побудував чат-бота який може направити абітурієнта до потрібного місця або відповісти йому на запитання.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Yoav Goldberg , Graeme Hirst, Neural Network Methods in Natural Language Processing
2. Ehud Reiter, Building Applied Natural Language Generation Systems
3. Ian Goodfellow, Yoshua Bengio and Aaron Courville
4. Daniel Jurafsky, James H. Martin, Speech and Language Processing
5. https://pathmind.com/wiki/lstm [електронний ресурс]
6. https://medium.com/sciforce/a-comprehensive-guide-to-natural-language-generation-dd63a4b6e548 [електронний ресурс]
7. https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf [електронний ресурс]
8. https://www.retresco.de/en/nlg-industries-formats/ [електронний ресурс]
9. https://www.articleforge.com/how-it-works [електронний ресурс]

10. https://medium.com/analytics-vidhya/regular-expressions-an-excellent-tool-for-text-analysis-or-nlp-d1fa7d666cb9 [електронний ресурс]
11. https://sigmoidal.io/boosting-your-solutions-with-nlp/ [електронний ресурс]
12. https://www.ibm.com/cloud/watson-assistant/ [електронний ресурс]
13. https://aws.amazon.com/ru/lex/ [електронний ресурс]
14. https://dev.botframework.com/ [електронний ресурс]
15. https://dialogflow.com/ [електронний ресурс]
16. https://cloud.google.com/dialogflow/docs/ [електронний ресурс]